UNIVERSITY OF ILLINOIS

May 22, 1984

THIS IS TO CERTIFY THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

KARL ROBERT KRAUSE

ENTITLED SIMULTANEOUS SOLUTION OF SYSTEMS OF INTERLINKED MULTISTAGED SEPARATORS USING THE CRAY X-MP SUPERCOMPUTER

IS APPROVED BY ME AS FULFILLING THIS PART OF THE REQUIREMENTS FOR THE

DEGREE OF BACHELOR OF SCIENCE IN CHEMICAL ENGINEERING

Mark Stadtherr

Instructor in Charge

APPROVED: Richard C. Alkire

HEAD OF DEPARTMENT OF CHEMICAL ENGINEERING

O-1364

Simultaneous Solution of Systems of Interlinked Multistaged Separators Using the Cray X-MP Supercomputer

By

Karl Robert Krause

Thesis

for the
Degree of Bachelor of Science
in
Chemical Engineering

College of Liberal Arts and Sciences
University of Illinois
Urbana, Illinois

1988

Contents

List of Figures

List of Tables

1 Introduction

Distillation is one of the most widely used industrial chemical processes and the modelling of simple distillation systems has received much attention. More complicated systems with interlinking sidestreams are frequently the most cost effective, however, but are also more difficult to simulate. The traditional method for simple systems has been to simulate one column, use the product streams from this column as the feed streams to the second column, simulate the second column, and so on until all of the columns have been solved. This sequential modular approach works well for simple systems but is not very efficient for interlinked systems because a sidestream from one column may affect a previously solved column. As a result, a large number of iterations over all of the columns is required to solve the system.

A more efficient solution method is to solve the entire system simultaneously. This technique works well with interlinked systems, but requires the solution of a large number of nonlinear equations. Each plate is modelled by 2C + 1 equations, where C is the number of components. Therefore, over 1,000 nonlinear equations must be solved simultaneously to simulate a typical system of 5 components and 100 plates. While advances in computer technology have allowed this technique to be used on computers such as the CYBER 175, the development of supercomputers is important because 1) their increased memory allows larger problems to be solved and 2) their vector processing capabilities can be used to improve the efficiency of existing programs.

The second aspect has been investigated in this report. The performance of existing programs[1] on the CYBER and the Cray X-MP supercomputer has been compared. Both computers were located at the University of Illinois-Urbana campus. The total solution time on the Cray was less than on the CYBER because of the faster clock speed of the Cray. The programs were then run on the Cray with the vectorization both on and off. Any difference in solution time was due only to the effect of vectorization. Finally, the CPU time spent in each DO loop and subroutine was determined to identify the locations in the code where vectorization would be the most beneficial.

The efficient solution of distillation problems is important as computer resources become more valuable. Also, flowsheeting programs are finding increased use in industry. The distillation columns frequently require the most time to simulate accurately. Significant reductions in the solution time for the distillation system, therefore, will improve the overall performance of the flowsheeting program.

2 Simultaneous Solution of Interlinked Distillation Columns

The equations that must be solved are nonlinear and successive linearization was used to solve the system. This involves approximating the nonlinear system as a linear system, solving the linear system, and using this new solution to relinearize the nonlinear system. The solution of the linear system converges to the solution of the nonlinear system.

2.1 Equation Formulation

The Naphtali-Sandholm equation formulation was used for most of the problems[2]. In this formulation, each stage is represented by 1 energy balance, C mass balances, and C equilibrium equations:

1 energy balance:

$$E_n \quad (1 + S_n)H_n + (1 + s_n)h_n \quad H_{n+1} \quad h_{n-1} \quad h_{f\,n}$$

C mass balances:

$$M_{n,m} \quad (1 + S_n)V_{n,m} + (1 + s_n)L_{n,m} \quad V_{n+1,m} \quad L_{n-1,m} \quad f_{n,m}$$

C equilibrium equations:

$$Q_{n,m} = (nK_{n,m}V_nL_{n,m})/L_n \quad V_{n,m} + (1 \quad n)V_{n+1,m}V_n/V_{n+1}$$

N number of plates, n 1, 2, ... N, m 1, 2, ... C

Figure 1 shows a generic plate and defines the variables.

The equilibrium K-values and the enthalpies are functions of temperature. The independent variables are NC vapor flow rates, NC liquid flow rates, and N plate temperatures. The E_n, $M_{n,m}$, and $Q_{n,m}$ are residuals which will equal zero when the system has been solved.

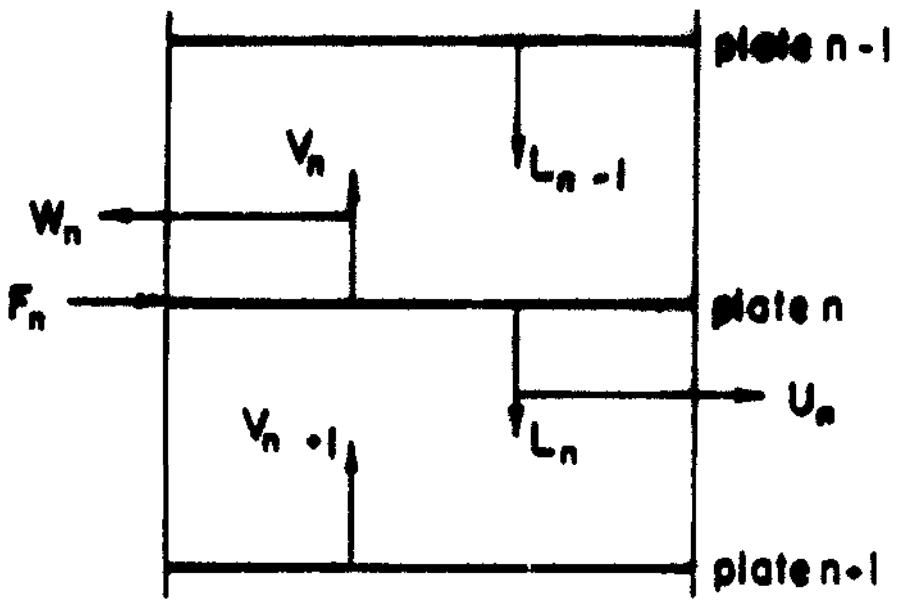
An alternative formulation4 involves summing the C equilibrium equations:

$$Z_n \quad L_n \quad \sum_{m=1}^{C} K_{n,m}L_{n,m}$$

and using this to replace one of the equilibrium equations. This formulation was used with the BBTF linear equation solver and is indicated by BBTF*.

Figure 1: Generic Distillation Column Plate

Including Nomenclature[4]



n = plate number

m = component number

L_n = liquid flow rate from plate n

V_n = vapor flow rate from plate

U_n = liquid sidestream from plate n

W_n = vapor sidestream from plate n

F_n = feed rate to plate n

2.2 Nonlinear Equation Solvers

The term 'solver' is actually a misnomer because the nonlinear equation solver is used to approximate the actual system with a linear system. This linear system can then be directly solved. Four different linear equation solvers were used.[5]

The Newton-Raphson (N-R) method is the classic method. The nonlinear sytem is expanded in a Taylor series about the current approximation to the solution, x, and all partial derivatives higher than first order are dropped. The resulting linear system can be represented by $A^{(k)}p^{(k)} = f^{(k)}$ where A is the Jacobian matrix of partial derivatives, p is the vector of variable corrections, and f is the set of nonlinear equations. The superscript (k) refers to the iteration number. The new estimate of the solution is $x^{(k+1)} = p^{(k)} + x^{(k)}$. The N-R method converges faster than the other methods but the Jacobian must be reevaluated at each iteration.

The Jacobian evaluation frequently requires more CPU time than solving the linear system. This is due to calls to time consuming thermodynamic property evaluation routines. The other methods, known as quasi-Newton methods, approximate or hold the Jacobian constant for several iterations. This reduces the time for each iteration but results in slower convergence and more iterations. The simplified Newton-Raphson (simp N-R) method simply holds the Jacobian constant for several iterations. The modified Broyden (MB) method updates the Jacobian without reevaluating the thermodynamic partial derivatives. Also, the linear system doesn't have to be resolved at each iteration. Schubert's (Schubert) method approximates the Jacobian in a way that preserves the sparsity of the matrix, but the linear system must be resolved at each iteration. The method of Dennis and Marwil (MARWIL) method essentially updates the Jacobian in a way that makes the solution of the linear system trivial. The drawback is that more information about the linear system is required. MARWIL is most efficient when used with the GENSOL linear equation solver.

2.3 Linear Equation Solvers

The methods of solving the linear system differ in the data structure used and in the matrix block structure they take advantage of.[6] The solvers used were continuous back substition (CBS), bordered-block-triangular form (BBTF), bordered-block-tridiagonal form (BBTDF1), LU factorization (LU1P9), Harwell subroutine library method MA28 (MA28A), sparse Gaussian elimination with partial pivoting (NSPIV), and implicit back substitution (RANK1). BBTDF1 stores each block of nonzero elements as a full matrix and is referred to as having a 1*tau storage scheme, where tau is the number of nonzero elements. NSPIV, LU1P9, RANK1, and CBS have 2*tau data structures, where each nonzero element and its column number are stored along with a small array of row locator indices. A 3*tau data structure is used by MA28A. Each nonzero element is stored along with its row and column numbers.

3 Comparison of Performance on the CYBER and the Cray

The programs were tested on 9 problems with 5 nonlinear equation solvers, 8 linear equation solvers, and 3 thermodynamic property evaluation routines. The problem specifications are summarized in Table 1 and detailed in reference 1.

Table 1: Problem Specifications

problem	components	stages	equations	columns	feeds	sidestreams
1	3	22	154	2	1	3
2	4	45	405	2	3	1
3	4	90	810	2	3	1
4	10	37	777	2	1	3
5	15	25	775	2	1	1
6	6	95	1235	3	1	3
7	6	120	1560	2	3	1
8	3	300	2100	2	2	4
9	3	330	2310	3	1	2

3.1 Differences Between CYBER and Cray FORTRAN

Originally, the programs were written in FORTRAN 4 on the CYBER. The Cray uses CFT, an extended version of FORTRAN 77, which has some different syntaxes than FORTRAN 4.[7] Minor changes in the program statement were necessary. Also, the syntax of the RETURNS statement was different. RETURNS allows a subroutine to return to different locations in the calling routine depending on results in the subroutine.

The CYBER does not distinguish between upper and lower case letters and the programs and data files were all upper case. The Cray uses both cases. The programs initially did not execute correctly on the Cray. The problem was found to be in the way character data was being read from the data file. The terms 'LIQ' and 'VAP' were used to designate liquid and vapor feed streams, respectively, but the Cray was reading 'LIq' and 'VAp'. Since these are not equivalent on the Cray, the programs were not calculating the stream properties correctly. The programs ran properly after changing all letters to lower case in both the data and FORTRAN files. The reason that the last letter being read as lower case has not been determined.

Another surprise occured when comparing runs on the Cray with the vectorization on (Cray-on) and off (Cray-off). In most cases the solution was reached slightly faster, as expected, by Cray-on than by Cray-off. Several runs, however, took 1 or 2 more iterations

with the vectorization on resulting in Cray on taking more CPU time to converge. Even when both methods converged in the same number of iterations the final sums of squares were slightly (up to 10%) different. While differences in the internal precision of the CYBER and the Cray are expected to produce slightly different results after several hundred thousand operations, calculations on the Cray should produce exactly the same result regardless of whether the vectorization is on or off. Slight numerical differences between matrix elements could result in different pivot selections, which could lead to a different number of iterations.

3.2 Total Solution Time

The average solution time for each problem is summarized in Table 2 for both numerical and analytical derivatives. The total solution times for each run on the CYBER, Cray-on, and Cray-off are listed in Tables 8 to 16. The Cray offers two advantages that allow programs to execute faster. The first is a clock speed that is about 5 times faster than conventional computers. This means that each numerical operation can be performed five times faster on the Cray. The second advantage is the ability to process large vector loops simultaneously. This is refered to as vectorization. Instead of iterating through a DO loop to update each element of an array or matrix, the Cray can update the entire matrix simultaneously. Code that is written to take advantage of vectorization can show a speedup of up to ten.[8] Speedup is defined, in this case, as the ratio of solution times Cray-off/Cray-on. Highly vectorized programs on the Cray can show a speedup of up 50 when compared to the CYBER due to both the faster clock speed and vectorization.

The column labeled 'runs' in Table 2 indicates the number of combinations of equation solvers and thermodynamic property derivatives used to solve a particular problem. Only runs where results were available from both the CYBER and the Cray were included in the tables to make the comparison between machines valid.

Comparison between solution times for CYBER and Cray-off indicate an average speedup of only 2.5, varying between 1.6 and 3.7 from problem to problem. This speedup is due to the faster clock speed only. One possible explanation for the less than expected increase is the memory structure of the Cray. The Cray memory is organized into 32 'banks.' Reading from a memory location in a given bank makes that bank unavailable for four clock cycles.[9] A 'bank conflict' occurs when an unavailable memory location is accessed and the number of bank conflicts depends on the dimensions of the array variables. Severe bank conflicts can reduce the efficiency by up to a factor of four. The variation in average solution times from problem to problem support the possibility of bank conflicts because each problem involves a different number of components, stages, and variables. Therefore, the arrays will have different dimensions.

Table 2: Comparison of Average Solution Time

numerical thermodynamic property derivatives

		average solution time (sec)			
problem	run	CYBER	Cray-off	Cray-on	speedup
1	7	6.12	2.533	2.472	1.02
2	7	23.9	8.649	7.675	1.13
3	7	70.4	18.907	18.788	1.01
4	1	80.9	49.601	49.331	1.01
5	4	105.	47.167	43.347	1.09
6	2	53.6	23.016	22.494	1.02
7	3	125.	47.410	46.084	1.03
8	1	51.9	24.954	24.769	1.01
9	2	68.7	32.922	31.104	1.06
average speedup from CYBER to Cray: 2.4					

analytical thermodynamic property derivatives

		average solution time (sec)			
problem	run	CYBER	Cray-off	Cray-on	speedup
1	8	---	1.667	1.611	1.03
4	2	38.8	15.610	16.033	
5	4	49.2	31.385	28.829	1.09
6	2	25.5	11.548	11.442	1.01
7	3	61.0	21.042	20.570	1.02
8	1	21.3	7.976	7.922	1.01
9	2	37.7	14.185	13.987	1.01
average speedup from CYBER to Cray: 2.4					

Very little speedup occured when the vectorization was turned on. This was expected because the code was not written to take advantage of this feature. Problem five showed the greatest improvement in solution time with the vectorization on and had the largest number of components. While the improvement was not very great, it could indicate that more loops involving the number of components were vectorized than loops that iterate on the number of plates.

3.3 Nonlinear Equation Solvers

The nonlinear equation solvers are compared in Table 3. Again, very little reduction in CPU time was noticed with the vectorization on. Problems 1 and 9 were run using each nonlinear equation solver with the same linear equation solver. Only differences in the nonlinear equation solvers accounted for the differences in solution times. All of the runs executed faster when analytical derivatives where used but the Newton-Raphson method, which reevaluates the Jacobian at each iteration, improved the most. Significant improvements in the vectorization of the Jacobian and physical property evaluation subroutines could greatly improve the efficiency of this method. Schubert's method and the Newton-Raphson method solve the problems faster than the other methods.

3.4 Linear Equation Solvers

The linear equation solvers are compared in Tables 4 and 5. The earlier results from the CYBER are not included for clarity but may be found in Table 17. The execution time for all of the linear equation solvers except CBS decreased by a factor of 4 or greater when run on the Cray. This was significantly larger than the average decrease in total solution time. There were no significant improvements in solution time when the vectorization was turned on except for BBTDF1 with a speedup of 1.14 This routine also executed 10 to 15 times faster on the Cray without vectorization than on the CYBER. This could indicate that the data was structured to avoid bank conflicts. BBTDF1 and LU1P9 showed the best overall performance.

3.5 Jacobian and Function Evaluations

Finally, neither the function nor the Jacobian evaluation routines were greatly affected by vectorization. Table 6 compares the times for the Cray and the CYBER. The speedup was the same for both derivative methods indicating that neither method had a large degree of vectorization. Numerical derivative calculations require about 5 times more CPU time than analytical calculations but may be easier to vectorize. These results further show that significant improvement in efficiency could be achieved by rewriting the programs to take advantage of vectorization.

Table 3: Comparison of Nonlinear Equation Solvers

Problem 1: 154 equations, 22 stages, 3 components
CBS linear equation solver, Chao-Seader correlation

nonlinear equation solver	thermodynamic property derivatives	total solution time (sec)			
		CYBER	Cray-off	Cray-on	speedup
N-R	numerical	3.70	1.854	1.841	1.01
M-B	numerical		2.123	2.099	1.01
Simp. N-R	numerical		2.275	2.245	1.01
Schubert	numerical	--	1.476	1.445	1.02
N-R	analytical	--	0.652	0.644	1.01
M-B	analytical		1.163	1.140	1.02
Simp. N-R	analytical		1.316	1.298	1.01
Schubert	analytical	--	0.995	0.974	1.02

Problem 9: 2310 equations, 330 stages, 3 components
LU1P9 linear equation solver, Chao-Seader correlation

nonlinear equation solver	thermodynamic property derivatives	total solution time (sec)			
		CYBER	Cray-off	Cray-on	speedup
N-R	numerical	65.7	32.078	31.951	1.00
M-B	numerical		44.150	43.555	1.01
Simp. N-R	numerical	--	42.069	41.530	1.01
Schubert	numerical	71.6	33.765	30.256	1.12
N-R	analytical	25.9	9.860	9.776	1.01
M-B	analytical	--	21.940	21.533	1.02
Simp. N-R	analytical		19.789	19.513	1.01
Schubert	analytical	49.4	18.509	18.197	1.02

Table 4: Comparison of Linear Equation Solvers

linear equation solver	time to solve linear system for Cray-on (sec) problem number								
	1	2	3	4	5	6	7	8	9
CBS	.036	.13	.52	2.9	3.1	3.0	1.9		
LU1P9	.020	.048	.103	.66	.71	.38	.32	.31	.25
RANK1	.028	.19	.73	1.5	1.3		2.7		
NSPIV	.021	.051		–	.74	.45	.37	.32	
BBTDF1	.017	.057	.117	.23	.33				
MA28A	.020	.9	2.8		–			–	–
BBTF*		.028	.052	–			–	–	
GENSOL	.029	–	–	–	–			–	

Table 5: Comparison of Vectorizability of Linear Equation Solvers

linear equation solver	number of problems	average CPU time to solve linear system (sec) CYBER	Cray-off	Cray-on	speedup
CBS	7	2.63	1.644	1.648	—
LU1P9	9	1.26	0.315	0.311	1.01
RANK1	6	3.42	1.069	1.082	—
NSPIV	2	0.19	0.036	0.036	1.00
BBTDF1	5	2.50	0.171	0.150	1.14
MA28A	3	4.18	1.274	1.240	1.03
BBTF*	2	3.12	0.040	0.040	1.00
Use to compare vectorizability of each linear equation solver–can't compare equation solvers because different different problems were used					

Table 6: Comparison of Time for Jacobian Evaluations and Function Evaluations

	thermodynamic property derivatives	average CPU time for evaluation (sec)			
		CYBER	Cray-off	Cray-on	speedup
Jacobian evaluation	numerical	6.01	2.296	2.241	1.02
	analytical	1.49	0.515	0.502	1.03
function evaluation	both	0.30	0.120	0.117	1.03

4 Opportunities for Improved Performance

The results clearly indicate that the programs are not written for efficient execution of the Cray. The execution time was broken done at the subroutine and the DO loop level by using the TIMER/TALLY utility. The results are promising in that a large portion of the execution time is spent in a small area of the code. While it is impossible to vectorize every DO loop, vectorization of small portions of the code could result in large speedups.

4.1 Breakdown of CPU Time by Subroutine

Problem 4 was run using the TIMER/TALLY utility and the results are summarized in Table 7. The highest percentage of time was spent in the linear equation solver subroutines regardless of the nonlinear equation solver used. The percentage increased, up to 91% for CBS, when analytical thermodynamic derivatives were used because less time was spent evaluating the Jacobian. Further analysis at the DO loop level indicated that over 80% of the total solution time was being spent in one DO loop within the subroutine CBSLU1. If this loop can be vectorized to achieve a speedup of 10 within the loop, the overall program will show a speedup of up to 8.

Table 7: Breakdown of Execution Time by Subroutine

Problem 4 run with Cray-on

nonlinear equation solver	linear equation solver	percentage of time in subroutine thermodynamic property derivatives			
		numerical		analytical	
N-R	NSPIV	NSPIV1	28%	NSPIV1	69%
		VLE1O	4%	JACOB2	6%
Schubert	LU1P9	LU1P2	64%	LU1P2	74%
		SCHUB	2%	SCHUB	3%
N-R	CBS	CBSLU1	60%	CBSLU1	91%
		VLE1O	2%	JACOB2	1%
Schubert	RANK1	RANK1	81%	RANK1	90%
		VLE1O	1%	SPK2	1%
M-B	BBTDF1	SOLVE2	20%	SOLVE2	23%
		INVERT	5%	INVERT	9%
less than 2% of all operations were vectorized					

TIMER/TALLY also reports the number of operations that were vectorized and the

number that were scalar. Less than 2% of the operations were vectorized. This is not surprising considering the above results.

4.2 Vectorization of Linear Equation Solvers

The possibility of vectorizing the CBSLU1 subroutine was examined. The portion of code where the majority of the time was being spent is shown in the top of Figure 2. Only the innermost loop can be vectorized and a loop with an IF statement in it can not be vectorized. Also, a loop can not alter an array value that would be used in a latter iteration if the loop was not vectorized. The innermost loop, DO 520 ..., in the original code was not vectorized for these and other reasons.

The total solution time was reduced by 2 seconds by rewriting the code as three separate loops and using a temporary array. The loop with label 527 in the bottom portion of Figure 2 is vectorized, but the loop with label 525 still is not. A more advanced understanding of the intricacies of vectorization would probably show that the whole loop can be vectorized, but this small modification confirms that the programs can be more efficient.

4.3 Vectorization of Other Subroutines

The thermodynamic property subroutines are called many times during execution, particularly when the N-R nonlinear equation solver is used. These would be likely candidates for further study. A point of diminishing returns is reached quickly, however, because very little time is spent in most of the subroutines. Completely vectorizing a subroutine that only takes 1% of the total time will only result in a speedup of about 1.1.

4.4 Alternative Solution Strategies

Perhaps the most promising approach would be an entirely different solution strategy. A more efficent algorithm written with vectorization in mind might give better results than patching up programs that were designed for scalar operations. Also, the virtually unlimited memory capacity of the Cray could be utilized in a way that allows for an efficient vectorized algorithm.

Figure 2: Revision of Code to Improve Vectorization

Subroutine CBSLU1 was revised to improve the vectorization of one DO loop. Over 80% of the total solution time was spent in this subroutine when problem 4 was solved with the Newton-Raphson nonlinear and the CBS linear equation solvers using analytical derivatives.

Original Code: 1% of all operations were vectorized, solution time = 15.995 sec

```
      IF (NSPK .EQ. 0) GOTO 540
      DO 530 NS = 1, NSPK
         JJP = CLI(JSPK(NS))
         DO 520 JJ = JJF, JJL
            IF (A(JP+JJ) .EQ. 0.) GOTO 520
            NOP = NOP + 1
            IF(A(JJP+JJ) .EQ. 0.) NFILL = NFILL + 1
            A(JJP+JJ)    A(JJP+JJ) - A(JJP+II)*A(JP+JJ)
520      CONTINUE
530   CONTINUE
```

Modified Code: 4% of all operations were vectorized, solution time = 13.957 sec

```
      DO 523 JJ = JJF, JJL
523      ATEMP(JJ) = 0.
      IF (NSPK .EQ. 0) GOTO 540
      DO 530 NS = 1, NSPK
         JJP = CLI(JSPK(NS))
         IF (A(JJP+II) .EQ. 0.) GOTO 530
         DO 525 JJ = JJF, JJL
            IF (A(JP+JJ) .EQ. 0.) GOTO 525
            NOP = NOP + 1
            IF (A(JJP+JJ) .EQ. 0.) NFILL = NFILL + 1
            ATEMP(JJ) = A(JJP+II)*A(JP+JJ)
525      CONTINUE
         DO 527 JJ = JJF, JJL
            A(JJP+JJ) = A(JJP+JJ) - ATEMP(JJ)
527         ATEMP(JJ) = 0.
530   CONTINUE
```

5 Acknowledgements

The guidance and patience of Dr. M.A. Stadtherr was essential throughout this project. The members of Professor Stadtherr's research group were also very helpful. Finally, the inspiration and motivational support provided by Julie Staley was greatly appreciated.

6 Bibliography

The following references were used in this report:

1. Malachowski, Mike., 'Simultaneous Solution of Multistaged Interlinked Separators,' Ph.D. Thesis, Univ. of Illinois, Urbana, Illinois (1983)
2. Ibid, pp. 37-41.
3. Ibid, p. 28.
4. Ibid, p. 43.
5. Ibid, pp. 86-95.
6. Ibid, pp. 59-86.
7. Cray Research Inc., 'Cray-1 and Cray X-MP Computer Systems. FORTRAN (CFT) Reference Manual,' revision J-02 (1984), p. 1-1.
8. National Center for Supercomputing Applications, 'User's Guide,' version 1.0, (January,1986), p. 10-7.
9. Ibid, p. 10-9.

A Appendices

A.1 Sample Program Output


```
 * * *  SOLUTION OF SYSTEM OF NONLINEAR EQUATIONS.

 DIAG = 2 (DEFAULT VALUES GIVEN)

 BROYDEN METHOD = 1 (USE SPARSE BROYDEN OPTIONS)

 PRINT CONTROL (NSLOT) OPTION (0 = NO INTERMEDIATE PRINTING)

 WORK STORAGE INFORMATION
 NUMBER OF EQUATIONS          1235
 NUMBER OF EQUIVALENCE CHANGES =        95
 NUMBER OF EQUIVALENCES FOR STORAGE =        11

 LIN =       0 (0 = NO LINEAR EQUATIONS 1 = LINEAR EQUATIONS)
 LIN1 =      1 (0 = CHANGE TO 1DLI ARRAY
 LIN2 =      1 (0 = 1DNI ARRAY FOR SINGLE SOLUTION METHOD
 LIN3 =      1 (0 = 1DN3 ARRAY FOR MULTIPLE SOLUTIONS WITH SAME PIVOT ORDER

 SCALE = 0 (0 = NO VARIABLE SCALING, 1 = VARIABLE SCALING)
 SCALE = 0 (0 = NO ROW SCALING, 1 = ROW SCALING)

 DEBUG FOR SOLUTION= 1 (0 = NO DEBUGGING OF SOLUTION, 1 = DEBUG SOLUTION)
 DEBUG FOR PRINTING OF INITIAL GUESS= 0
 DEBUG FOR PRINTING OF FINAL ESTIMATE OF SOLUTION= 0
 DEBUG FOR PRINTING OF JACOBIAN MATRIX= 0

 MINIMUM REQUIRED FOR ELEMENT ARRAYS:       100 K
 ESTIMATED ACC REQUIRED FOR ARRAY WORK:       4200
 TOTAL SPACE REQUIRED=      43830
 TOTAL SPACE ALLOCATED=     130000
 SPACE USED FOR SPARSE WORK=       41111

 RELSYM (RELATIVE SYMBOLIC ZERO) =  0.100E-02
 RELDIF (RELATIVE DIFFERENCE) =  0.100E-02    (UNUSED)
 RELSOL (RELATIVE SOLUTION) =  0.100E-02    (UNUSED)
 EPSMCH (MACHINE EPSILON) OBTAINED=  0.190E-06

 MAXIMUM NUMBER OF COLUMN INTERCHANGES =       50          COLUMN SCALING OPTION =  0

 MAXIMUM ITERATION COUNT =  1

 MAXIMUM ALLOWABLE STEP SIZE FOR NEWTON ITERATION =        0.00
 MINIMUM PERMISSIBLE LIMIT FOR PIVOTAL ELEMENT =        0.00

 ITERATION NO.=  0  PLOT  NORM OF F(X) =  0.

 NUMBER OF ITERATIONS IN  MULTIGRID CORRECTION    NUMERICAL DERIVATIVES

 VARIABLE     INITIAL GUESS      RIGHT HAND SIDE      GENERIC FACTOR      SCALED EST. PARAMETER      CORR. FACTOR
    1      0.500000E+03      0.240000E+00      0.178400E+00      0.200000E+00      0.000000E+00
    2      0.952500E+03      0.375000E+02      0.149300E+00      0.218420E+00      0.123000E+02
    3      0.960400E+03      0.234000E+00      0.112700E+00      0.211000E+00      0.179000E+02
    4      0.942600E+03      0.133000E+02      0.975100E+00      0.200000E+00      0.000000E+00
    5      0.750000E+03      0.293000E+02      0.294000E+00      0.173000E+00      0.117600E+02
    6      0.750000E+03      0.270000E+00      0.975100E+00      0.217000E+00      0.179000E+02

 VARIABLE                    VARIABLE VALUE CORRECTIONS
    1=0.000000E+00  -0.111001E 00   0.918990E-04  -0.999090E-07   0.699110E 11
    2=0.000000E+00  -0.224405E 00   0.791300E-04  -0.284970E-07  -0.674000E-11
    3=0.000000E+00  -0.227501E 00   0.121000E-03  -0.871170E-07   0.136700E-10
    4=0.000000E+00  -0.116400E-03   0.116900E-03  -0.816300E-07   0.124700E-10
    5=0.000000E+00   0.105100E-01   0.137000E-03  -0.794900E-07   0.155100E-10
    6=0.000000E+00  -0.111900E 01   0.181500E-01  -0.872000E-07   0.150900E-10
```

GAS CONSTANT R = 0.19870E-01
MATRIX CONTAINS 4 OFF-BAND BLOCKS
A-MATRIX AT POSITION (2, 95) INTERLINK SPLIT FRACTION = 0.10
A-MATRIX AT POSITION (28, 31) INTERLINK SPLIT FRACTION = 0.10
A-MATRIX AT POSITION (61, 70) INTERLINK SPLIT FRACTION = 10.00
A-MATRIX AT POSITION (76, 45) INTERLINK SPLIT FRACTION = 10.00

MATRIX CONTAINS 4 ZERO RANK BLOCKS
ZERO MATRIX AT POSITION (30, 31)
ZERO MATRIX AT POSITION (31, 30)
ZERO MATRIX AT POSITION (65, 66)
ZERO MATRIX AT POSITION (66, 65)

COLUMN PRESSURE = 0.10000E+01

FEED NUMBER 1 FEED TO PLATE NUMBER 66 FEED CONDITION = LIQUID FEED TEMPERATURE = 305.00
COMPONENT FLOW RATES 0.50000E-01 0.50000E-01 0.90000E+00 0.90000E+00 0.10000E-01

HEAT EXCHANGER NUMBER 1 HEAT LOAD AT PLATE NUMBER 1 = 0.00000E+00
HEAT EXCHANGER NUMBER 2 HEAT LOAD AT PLATE NUMBER 30 = 0.00000E+00
HEAT EXCHANGER NUMBER 3 HEAT LOAD AT PLATE NUMBER 31 = 0.00000E+00
HEAT EXCHANGER NUMBER 4 HEAT LOAD AT PLATE NUMBER 65 = 0.00000E+00
HEAT EXCHANGER NUMBER 5 HEAT LOAD AT PLATE NUMBER 66 = 0.00000E+00
HEAT EXCHANGER NUMBER 6 HEAT LOAD AT PLATE NUMBER 95 = 0.00000E+00

REFLUX RATIO FOR PLATE 1 = 0.1000E+01

BOTTOMS RATE FOR PLATE 30 = 0.1000E+01

REFLUX RATIO FOR PLATE 31 = 0.1000E+01

BOTTOMS RATE FOR PLATE 65 = 0.1000E+01

REFLUX RATIO FOR PLATE 66 = 0.1000E+01

SIDESTREAM NUMBER 1 SIDESTREAM WITHDRAWN FROM PLATE NUMBER 95
SIDESTREAM CONDITION = LIQ SIDESTREAM SPLIT FRACTION = 0.0100

FLASH FUNCTION EVALUATION 0.100 SECONDS

TOTAL POWER REQUIREMENT = 0.21810

ITERATION NUMBER 1

TOTAL COST

NUMBER OF DIAGONAL BLOCKS 117
DIAGONAL BLOCKS 7761
UPPER OF DIAGONAL BLOCKS 117
L FACTOR BLOCKS 0
U FACTOR BLOCKS 0
TOTAL 1164

TIME FOR MATRIX BOOKKEEPING 0.191 SECONDS

TIME FOR DECOMPOSITION EVALUATION OR UPDATE 0.211 SECONDS

NUMBER OF ITERATIONS 1
NUMBER OF FEED IN 1
TOTAL OPERATIONS FOR SOLUTION =
STORAGE FOR SOLUTION OF STAGE #

```
ARRAY STORAGE FOR INVERSE =          8312

TIME FOR INVERSION AND SOLUTION=   0.496 SECONDS

 NORM OF RESIDUALS FOR LINEAR EQUATION SOLVER = 1.033649266434e1,922196[illegible]
 NORM OF RIGHT HAND VECTOR = [illegible]
 NORM OF SOLUTION VECTOR = 4.301197171501573639928425S1E+4

STEP SIZE =   0.1000E+01 SUM OF SQUARES =   0.510[illegible]

STEP SIZE =   0.1000E+00 SUM OF SQUARES =   0.125[illegible]

TIME FOR STEP SIZE CALCULATION =    0.[illegible] SECONDS

ITERATION NUMBER    3

TIME FOR JACOBIAN EVALUATION OR UPDATE =    2.[illegible] SECONDS

TIME FOR INVERSION AND SOLUTION=   0.458 SECONDS

STEP SIZE =   0.1000E+01 SUM OF SQUARES =   0.122589126467[illegible]

TIME FOR STEP SIZE CALCULATION =    0.109 SECONDS

ITERATION NUMBER    4

TIME FOR JACOBIAN EVALUATION OR UPDATE =    2.[illegible] SECONDS

TIME FOR INVERSION AND SOLUTION=   0.457 SECONDS

STEP SIZE =   0.1000E+01 SUM OF SQUARES =   0.[illegible]

TIME FOR STEP SIZE CALCULATION =    0.109 SECONDS

ITERATION NUMBER    5

TIME FOR JACOBIAN EVALUATION OR UPDATE =    2.[illegible] SECONDS

TIME FOR INVERSION AND SOLUTION=   0.452 SECONDS

STEP SIZE =   0.1000E+01 SUM OF SQUARES =   0.[illegible]

TIME FOR STEP SIZE CALCULATION =    0.109 SECONDS

SOLUTION HAS BEEN ATTAINED
SUM OF SQUARES =   0.[illegible]      IS LESS THAN CONVERGENCE CRITERION OF   0.1000E-09
TOTAL NUMBER OF ITERATIONS=    5      NUMBER OF JACOBIAN EVALUATIONS=    [illegible]
EXECUTION TIME =   16.[illegible] SECONDS
```

A.2 Sample TIMER/TALLY Output

HITS= 3919

INSTRUCTION TYPE	SCALAR %	VECTOR %
IMMED	43	0
MEMORY	9	35
A ADD	10	0
A MULT	0	0
POP/LI	0	0
LOGICAL	2	11
SHIFT	0	0
INT ADD	17	11
FP ADD	11	17
FP MULT	5	23
FP DIV	0	0
TOTAL HITS = 3919	97	0

LOCATION	LENGTH	SUBROUTINE	HITS	PERCENT	
00723306	00001651	CBLAU1	3550	90.6075	*
00665E00	00000105	$MULT	61	1.5569	9.3925
00714610	00002014	JACOB3	54	1.3783	7.8356
00671600	00000077	$ADD	49	1.2506	6.4573
00712757	00000464	SPEC	35	.8933	5.2067
00746E00	00000117	ALOG10	21	.5360	4.3134
00746672	00000446	ROOT	15	.3828	3.7774
00677600	00000085	FTOI%	11	.2808	3.4966
00746700	00000111	EXP	10	.2553	3.1158
00747162	00000072	IGENTH	7	.1787	2.9371
00700256	00001101	VLE10	6	.1532	2.7584
00777450	00000051	IGOT	5	.1276	2.6307
00756117	00000250	VLE11	5	.1276	2.5031
00760667	00000404	@PUILD	4	.1021	2.4010
00754E76	00001106	VLE10	4	.1021	2.2989
00633343	00000264	STORE2	4	.1021	2.1968
00777370	00000070	IGOT	3	.0766	2.1201
007E4047	00000E27	VLE1F	3	.0766	2.0435
00746340	00000116	ASET	3	.0766	1.9668
00702220	00000406	VLE11	3	.0766	1.8902
00671700	00000066	$DIV	3	.0766	1.8135
00755704	00000213	VLE1C	4	.1021	1.7369
00745600	00000072	%SORT	4	.1021	1.6348
00741273	00000405	CBUILD	3	.0766	1.5326
00712046	00000172	SETUP	3	.0766	1.4560
00617241	00001676	FUNCEV	3	.0766	1.3793
00756367	00000133	VLE1C	2	.0510	1.3027
00711217	00000172	APUILD	2	.0510	1.2517
00701357	00000641	VLE1M	2	.0510	1.2006
00677655	00000401	NFRAC	2	.0510	1.1496
00671153	00000324	@GET1	2	.0510	1.0985
00605300	00000361	$NEGV	2	.0510	1.0475
00753004	00001043	VLE1E	1	.0255	1.0217
00747100	00000063	RTOR%	1	.0255	1.0089
00713443	00000150	SPERFORM	1	.0255	.9962
00631200	00000780	JACOB4	1	.0255	.9834
00607035	00000554	@WRITEP	1	.0255	.9707
00604122	00000164	@77ERR1	1	.0255	.9579
00546531	00001671	$WFI	1	.0255	.9452

A.3 Subroutine Structure of the Programs

The program itself is SIM25 and is stored in the file sim290. The program performs some initialization and then calls the subroutine MAIN located in the file lcsim28i. Main then calls the following subroutines:

routine:	file:	routine:	file:	routine:	file:
DATA	lc28x	VSCALE	lc28x	PRINTF	lc28x
SCHUB0	lc28x	SCHUB	lc28x	BH	lc28x
BHA	lc28x	SIMP	lc28x	MARWIL	lc28g5
JACOB5	lc28b	JACOB3	lc28b	JACOB4	lc28b
PRINT1	lc28x	STORE3	lc28x	STORE2	lc28x
TALLY1	lcorder	TALLY3	lc28x	RSCALE	lc28x
FSCALE	lc28x	JACOB	lc28b	BBTF0	lc28g6
GENSOL	lc28g5	RANK1	lcrank	CBSLU	lccbslu
BBTF	lc28g4	MA28AA	lc28x	M28*BB	lc28c
LU1P9	lclu1p9	NSPIVC	lcnspiv1	SOLVE0	lc28g6
STORE4	lc28x	BBTFP	lc28x	SOLVE3	lc28g4
MA28CC	lc28x	SOLVEG	lc28g5	STEP	lc28x
PRINTR	lc28x	RESID1	lc28i	RESID0	lc28i

The thermodynamic function routines for Chao-Seader correlation with numerical derivatives are stored in lc21d4 and with analytical derivatives in lc21d. The Uniquac routines are stored in lcuniq. Most of the subroutines listed above call other subroutines located in the same file. The modified version of the subroutine CBSLU1 is stored in vcbs. Data files are stored as lcdXXX where XXX denotes the problem and linear equation solver used.

A.4 Total Solution Time of Each Run

The following tables list the total solution time for each run on the Cray. Data from the CYBER is included where available.

Table 8: Total Solution Time for Each Run for Problem 1

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	BBTDF1	4.00	3.686	4.019
N-R	GENSOL	10.5	3.568	3.577
Schubert	MA28A	8.38	1.436	1.474
N-R	RANKI	3.79	1.808	1.816
N-R	CBS	3.70	1.841	1.854
N-R	LU1P9	3.76	1.763	1.771
N-R	NSPIV	—	1.769	1.783
MARWIL	GENSOL	8.70	3.202	3.220
M-B	RANKI	—	2.069	2.096
M-B	LU1P9	—	2.013	2.036
M-B	CBS	—	2.245	2.275
Simp N-R	CBS	—	2.245	2.275
Schubert	CBS	2.96	1.445	1.476

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	BBTDF1	—	4.665	5.050
N-R	GENSOL	—	2.379	2.377
Schubert	MA28A	—	0.960	0.994
N-R	RANKI	—	0.614	0.618
N-R	CBS	—	0.644	0.652
N-R	LU1P9	—	0.570	0.575
N-R	NSPIV	—	0.576	0.581
MARWIL	GENSOL	—	2.482	2.490
M-B	RANKI	—	1.116	1.140
M-B	LU1P9	—	1.039	1.073
M-B	CBS	—	1.140	1.163
Simp N-R	CBS	—	1.298	1.316
Schubert	CBS	—	0.974	0.995

Table 9: Total Solution Time for Each Run for Problem 2

Uniquac correlation with numerical derivatives				
nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	BBTDF1	18.4	13.291	16.150
Schubert	MA28A	53.7	16.695	18.381
Simp N-R	BBTF*	13.2	4.226	4.115
N-R	CBS	13.8	3.077	2.970
Schubert	LU1P9	13.7	2.828	2.825
N-R	NSPIV	-	2.716	2.605
MARWIL	GENSOL	34.9	8.971	8.859
Schubert	RANKI	19.3	4.640	4.641

Table 10: Total Solution Time for Each Run for Problem 3

Uniquac correlation with numerical derivatives				
nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	LU1P9	40.4	8.704	9.389
N-R	CBS	47.6	12.101	11.736
N-R	RANKI	60.0	13.829	13.713
MARWIL	GENSOL	197.2	50.866	51.030
Simp N-R	BBTF*	48.4	16.867	16.412
Simp N-R	BBTDF1	37.2	14.219	15.066
Simp N-R	MA28A	61.7	14.927	14.984

Table 11: Total Solution Time for Each Run for Problem 4

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	LU1P9	63.5	18.741	17.980
N-R	CBS	55.1	23.877	24.408
Schubert	RANKI	103.4	31.433	29.473
Schubert	BBTDF1	101.5	123.273	126.633

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
Schubert	LU1P9	109.7	42.171	43.888
N-R	CBS	29.3	15.995	15.865
Schubert	RANKI	—	23.737	24.992
Schubert	BBTDF1	—	102.303	115.997

Table 12: Total Solution Time for Each Run for Problem 5

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	CBS	109.7	42.171	43.888
N-R	NSPIV		14.290	15.381
N-R	LU1P9	89.3	22.832	24.525
N-R	RANKI	107.1	27.366	29.161
N-R	BBTDF1	115.2	81.019	91.092

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	CBS	53.4	27.768	28.095
N-R	NSPIV	-	5.303	5.478
N-R	LU1P9	33.2	8.338	8.619
N-R	RANKI	50.9	13.019	13.580
N-R	BBTDF1	59.4	66.192	75.245

Table 13: Total Solution Time for Each Run for Problem 6

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec) CYBER	Cray-on	Cray-off
N-R	NSPIV		15.881	16.402
N-R	CBS	62.8	29.459	29.952
N-R	LU1P9	44.4	15.528	16.079

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec) CYBER	Cray-on	Cray-off
N-R	NSPIV		4.829	5.003
N-R	CBS	31.5	18.379	18.428
N-R	LU1P9	16.4	4.508	4.670

Table 14: Total Solution Time for Each Run for Problem 7

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	NSPIV	--	34.952	35.957
N-R	CBS	112.5	47.909	48.997
N-R	LU1P9	95.6	34.469	35.724
N-R	RANKI	167.1	55.874	57.510

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	NSPIV	-	9.327	9.642
N-R	CBS	47.9	22.345	22.721
N-R	LU1P9	31.7	8.929	9.279
N-R	RANKI	103.4	30.436	31.127

Table 15: Total Solution Time for Each Run for Problem 8

Chao-Seader correlation with numerical derivatives				
nonlinear equation solver	linear equation solver	total CPU time (sec) CYBER	Cray-on	Cray-off
N-R	NSPIV		24.836	24.979
N-R	LU1P9	51.9	24.769	24.954

Chao-Seader correlation with analytical derivatives				
nonlinear equation solver	linear equation solver	total CPU time (sec) CYBER	Cray-on	Cray-off
N-R	NSPIV		7.982	8.070
N-R	LU1P9	21.3	7.922	7.976

Table 16: Total Solution Time for Each Run for Problem 9

Chao-Seader correlation with numerical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	LU1P9	65.7	31.951	32.078
Simp N-R	LU1P9	—	41.530	42.069
M-B	LU1P9	-	43.555	44.150
Schubert	LU1P9	71.6	30.256	33.765

Chao-Seader correlation with analytical derivatives

nonlinear equation solver	linear equation solver	total CPU time (sec)		
		CYBER	Cray-on	Cray-off
N-R	LU1P9	25.9	9.776	9.860
Simp N-R	LU1P9	—	19.513	19.789
M-B	LU1P9	—	21.553	21.940
Schubert	LU1P9	49.4	18.197	18.509

A.5 Performance of Linear Equation Solvers on the CYBER

The results from reference 1 are given in the Table below.

Table 17: Solution Times for Linear Equation Solvers on the CYBER

	LINEAR EQUATION SOLVER	SOLUTION TIME (SECONDS)		LINEAR EQUATION SOLVER	SOLUTION TIME (SECONDS)
PROBLEM 1	CBS	.07	PROBLEM 2	LU1P9	.21
154 EQUATIONS	LU1P9	.09	405 EQUATIONS	CBS	.21
22 STAGES	RANKI	.10	45 STAGES	NSPIV	.27
3 COMPONENTS	NSPIV	.11	4 COMPONENTS	BBTDF2	.56
	BBTDF1	.16		BBTDF1	.59
	BBTDF2	.17		RANKI	.73
	BBTF*	.18		THOMAS	.84
	THOMAS	.26		BBTF*	.97
	BBTF	.36		BBTF	1.47
	MA28A	.67		MA28A	3.08
PROBLEM 3	LU1P9	.46	PROBLEM 4	LU1P9	2.44
810 EQUATIONS	CBS	.71	777 EQUATIONS	BBTDF2	3.18
90 STAGES	BBTDF2	1.13	37 STAGES	BBTDF1	4.42
4 COMPONENTS	BBTDF1	1.21	10 COMPONENTS	RANKI	4.43
	THOMAS	1.64		CBS	4.73
	RANKI	2.25		THOMAS	10.23
	BBTF*	5.26			
	BBTF	6.07			
	MA28A	8.78			
PROBLEM 5	LU1P9	2.81	PROBLEM 6	LU1P9	1.48
775 EQUATIONS	BBTDF2	3.62	1235 EQUATIONS	BBTDF2	4.23
25 STAGES	RANKI	4.86	95 STAGES	CBS	4.55
15 COMPONENTS	BBTDF1	6.10	6 COMPONENTS	BBTDF1	5.01
	CBS	6.11		THOMAS	6.12
				RANKI	6.91
PROBLEM 7	LU1P9	1.34	PROBLEM 8	LU1P9	1.34
1560 EQUATIONS	CBS	2.04	2100 EQUATIONS	BBTDF1	2.19
120 STAGES	BBTDF2	3.45	300 STAGES	BBTDF2	2.35
6 COMPONENTS	BBTDF1	4.03	3 COMPONENTS		
	RANKI	8.14			
PROBLEM 9	LU1P9	1.13			
2310 EQUATIONS	BBTDF1	2.42			
330 STAGES	THOMAS	3.01			
3 COMPONENTS					